International Journal of Approximate Reasoning 80 (2017) 313–333

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

A survey of lifted inference approaches for probabilistic logic programming under the distribution semantics ☆

Fabrizio Riguzzi [a,*], Elena Bellodi [b], Riccardo Zese [b], Giuseppe Cota [b], Evelina Lamma [b]

[a] Dipartimento di Matematica e Informatica – Università di Ferrara, Via Saragat 1, 44122, Ferrara, Italy
[b] Dipartimento di Ingegneria – Università di Ferrara, Via Saragat 1, 44122, Ferrara, Italy

ARTICLE INFO

Article history:
Received 22 January 2016
Received in revised form 5 October 2016
Accepted 6 October 2016
Available online 13 October 2016

Keywords:
Probabilistic logic programming
Lifted inference
Variable elimination
Distribution semantics
ProbLog
Statistical relational artificial intelligence

ABSTRACT

Lifted inference aims at answering queries from statistical relational models by reasoning on populations of individuals as a whole instead of considering each individual singularly. Since the initial proposal by David Poole in 2003, many lifted inference techniques have appeared, by lifting different algorithms or using approximation involving different kinds of models, including parfactor graphs and Markov Logic Networks. Very recently lifted inference was applied to Probabilistic Logic Programming (PLP) under the distribution semantics, with proposals such as LP^2 and Weighted First-Order Model Counting (WFOMC). Moreover, techniques for dealing with aggregation parfactors can be directly applied to PLP. In this paper we survey these approaches and present an experimental comparison on five models. The results show that WFOMC outperforms the other approaches, being able to exploit more symmetries.

1. Introduction

Statistical relational models [1,2] describe domains with many individual entities connected by uncertain relations. Reasoning with models of the real world is often very costly due to the complexity of the models. However, sometimes the cost of reasoning can be reduced by exploiting symmetries in the model. This is the task of "lifted" inference, that answers queries by reasoning on populations of individuals as a whole instead of considering each individual singularly. The exploitation of the symmetries in the model can significantly speed up inference.

Lifted inference was initially proposed by David Poole in 2003 [3]. Since then, many techniques have appeared, lifting algorithms such as variable elimination and belief propagation, using approximation and dealing with models such as parfactor graphs and Markov Logic networks [4–6].

Lifted inference was applied to Probabilistic Logic Programming (PLP) only very recently. The first work is [7], where the authors describe the Prolog Factor Language (PFL), a representation in Prolog of first-order probabilistic factor models. The authors also present an implementation of lifted variable elimination and lifted belief propagation for PFL.

In PLP, most languages are based on the distribution semantics [8], such as Probabilistic Horn Abduction [9], PRISM [10], Independent Choice Logic [11], Logic Programs with Annotated Disjunctions [12], and ProbLog [13,14]. Applying lifted infer-

☆ This paper is part of the virtual special issue on Probabilistic logic programming 2015, edited by J. Vennekens and F. Riguzzi.

* Corresponding author.
E-mail addresses: fabrizio.riguzzi@unife.it (F. Riguzzi), elena.bellodi@unife.it (E. Bellodi), riccardo.zese@unife.it (R. Zese), giuseppe.cota@unife.it (G. Cota), evelina.lamma@unife.it (E. Lamma).

http://dx.doi.org/10.1016/j.ijar.2016.10.002
0888-613X/

Table 1
Notation used in this paper.

Concept	Notation
Logical variable	Typewriter upper case letters X, Y, ...
Vector of logical variables	Typewriter bold case letters **X**, **Y**, ...
Constant	Typewriter lower case letters x, y, ...
Factor	Italic upper case letters or (if the context is clear) Greek letters X, ϕ, ...
Logical atom/predicate symbol, random variable (RV)	Italic upper case letters X, Y, ...
Value assigned to RV	Italic lower case letters x, y, ...
Vector of RVs	Bold italic upper case letters $\boldsymbol{X}$, $\boldsymbol{Y}$, ...
Value assigned to vector of (parameterized) RVs	Bold italic lower case letters $\boldsymbol{x}$, $\boldsymbol{y}$, ...
Parameterized random variable (PRV) or parfactor	Italic sans serif upper case letters X, Y, ...
Vector of PRVs	Bold italic sans serif upper case letters $\boldsymbol{\mathsf{X}}$, $\boldsymbol{\mathsf{Y}}$, ...
Set of constraints	Calligraphic $\mathcal{C}$
Code	Typewriter

ence to PLP languages under the distribution semantics (PLPDS) is problematic because the conclusions of different rules are combined with noisy-OR that requires aggregations at the lifted level when existential variables are present. For example, consider the following ProbLog program from [15]:

```
p :: famous(Y).
popular(X) :- friends(X, Y), famous(Y).
```

where p is a real value corresponding with the probability of the probabilistic fact. In this case $P(\texttt{popular(john)}) = 1-(1-p)^m$ where m is the number of friends of john. This is because the body contains a variable not appearing in the head, that is thus existentially quantified. A grounding of the atom in the head of this clause represents the noisy-OR (without *leak probability*) of a number of ground bodies. In this case we do not need to know the identities of these friends, we just need to know how many there are. Hence, we need not to ground the clauses.

An exhaustive survey about lifted inference was proposed in [16]. However its focus is on Statistical Relational Learning and probabilistic graphical models techniques in general and does not handle specifically existential variables and aggregation.

The first works applying lifted inference directly to PLPDS appeared in 2014. In [17] the authors proposed LP^2 (for Lifted Probabilistic Logic Programming) that answers queries to ProbLog by translating the program into PFL and using an extended GC-FOVE lifted variable elimination algorithm.

Weighted First Order Model Counting (WFOMC) [18] instead uses a Skolemization algorithm for model counting problems that eliminates existential quantifiers from a first-order logic theory without changing its weighted model count. As such, it can be applied to PLPDS.

Aggregation is also treated in [19] where the authors proposed an aggregation operator for first directed first-order models that is independent of the sizes of the populations, in order to handle contexts in which a parent random variable is parameterized by logical variables that are not present in a child random variable.

In this paper, we survey these three proposals and experimentally evaluate them. The results show that inference time linearly increases with the number of individuals of the domain for approaches exploiting lifted variable elimination, while it is constant in case of WFOMC, thus indicating that the latter is able to lift a larger portion of the model.

The paper is organized as follows. Section 2 introduces preliminaries regarding ProbLog, PFL, Causal Independence Variable Elimination, and GC-FOVE. Section 3 presents LP^2 and shows the translation of ProbLog into PFL. Section 4 illustrates the use of aggregation parfactors for ProbLog. Section 5 describes WFOMC. Section 6 discusses how to apply these algorithms to non-tight logic programs. Section 7 reports the experiments performed and Section 8 concludes the paper.

2. Preliminaries

2.1. Notation

Lifted inference techniques exploit concepts from relational logic and probabilistic theory. Unfortunately these two branches of mathematics sometimes use the same term to indicate different concepts. For example the word "variable" means logical variable in the context of relational logic, whereas it means random variable in the field of probabilistic theory. In order to avoid confusion, we use different fonts to represent different meanings. Table 1 shows the notation used throughout the paper.

2.2. ProbLog

ProbLog [13,14] is a PLP language with a simple syntax and can be considered as the prototype of PLPDS. A ProbLog program consists of a set of rules (a normal logic program) plus a set of *ground probabilistic facts*. Ground probabilistic facts